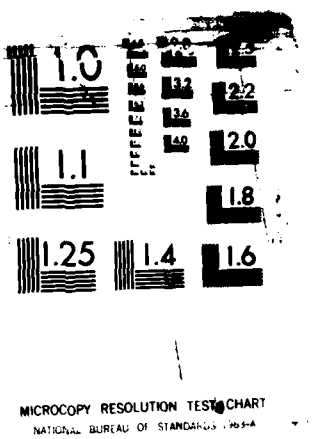
AD-A174 810 A SYSTEM FOR COMPUTER-AIDED DESIGN OF PRINTED CIRCUIT BOARDS(U) MATERIALS RESEARCH LABS ASCOT VALE (AUSTRALIA) T PIETSCH AUG 86 MRL-R-851-REV 1/1

UNCLASSIFIED F/G 9/1 NL

[redacted]

END



(12)

MRL-R-851

AR-003-032

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

MATERIALS RESEARCH LABORATORIES

MELBOURNE, VICTORIA

REPORT

MRL-R-851

A SYSTEM FOR COMPUTER-AIDED DESIGN OF PRINTED CIRCUIT BOARDS

T. Pietsch

Approved for Public Release

"Original contains colour plates: All DTIC reproductions will be in black and white"

APRIL, 1982
Revised AUGUST, 1986

DEPARTMENT OF DEFENCE
MATERIALS RESEARCH LABORATORIES

REPORT

MRL-R-851

A SYSTEM FOR COMPUTER-AIDED DESIGN OF PRINTED CIRCUIT BOARDS

T. Pietsch

ABSTRACT

A general purpose system for interactive computer-aided design of printed circuit boards is described. Included are details for encoding a circuit diagram in the computer, the placement of electronic component packages, the determination of a link/wire list and the routing of this list to obtain a wire layout. Design documentation in the form of component schedules and layout drawings are produced so that artwork can be generated on a Gerber drafting machine. A feature of the system is the method by which the routers find different size wire-paths on successive grids of increasing resolution. The use of this method approximately halves the routing time and increases the link connection rate by over 15%.

POSTAL ADDRESS: Director, Materials Research Laboratories
P.O. Box 50, Ascot Vale, Victoria 3032, Australia

SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED

DOCUMENT CONTROL DATA SHEET

REPORT NO.	AR NO.	REPORT SECURITY CLASSIFICATION
MRL-R-851	AR-003-032	Unclassified

TITLE

A system for computer-aided design of printed circuit boards

AUTHOR(S)

T. Pietsch

CORPORATE AUTHOR

Materials Research Laboratories
P.O. Box 50,
Ascot Vale, Victoria 3032

REPORT DATE	TASK NO.	SPONSOR
April, 1982 August, 1986, Revised		

FILE NO.	REFERENCES	PAGES
G6/4/8-2290	13	48

CLASSIFICATION/LIMITATION REVIEW DATE

CLASSIFICATION/RELEASE AUTHORITY

Superintendent, MRL
Physics Division

SECONDARY DISTRIBUTION

Approved for Public Release

ANNOUNCEMENT

Announcement of this report is unlimited

KEYWORDS

Printed circuit boards
Computer aided design
Computer graphics
Drafting
Computer programs

COSATI GROUPS 0901

ABSTRACT

A general purpose system for interactive computer-aided design of printed circuit boards is described. Included are details for encoding a circuit diagram in the computer, the placement of electronic component packages, the determination of a link/wire list and the routing of this list to obtain a wire layout. Design documentation in the form of component schedules and layout drawings are produced so that artwork can be generated on a Gerber drafting machine. A feature of the system is the method by which the routers find different size wire-paths on successive grids of increasing resolution. The use of this method approximately halves the routing time and increases the link connection rate by over 15%.

CONTENTS

CONTENTS
(continued)

CONTENTS
(continued)

A SYSTEM FOR COMPUTER-AIDED DESIGN OF PRINTED CIRCUIT BOARDS

1. INTRODUCTION

The manual design of printed circuit boards (pcb's) is a tedious and time-consuming occupation that requires meticulous attention to detail. A large board layout may take several months of design effort and, as this work is particularly subject to human error, careful checking procedures must be adopted throughout to avoid mistakes. As boards become more complex these problems are compounded and methods that inherently avoid human error whilst reducing design time are to be welcomed. The use of these methods should increase both the productivity and the realiability of pcb design.

Modern computers have the capacity to store and manipulate vast amounts of data efficiently and quickly. They are eminently suited to accept into memory the large amounts of detail involved in a board design so that data integrity can be maintained without continual cross-checking with the original circuit. Such cross-checking is usually prone to human error.

For these reasons a System for Computer-Aided Design of Printed Circuit Boards (SCAD.PCB) has been evolved at Materials Research Laboratories. It consists of a set of interrelated computer programs implemented in DEC-System 10/20 Algol 60 [1] and designed to handle every aspect of a board design in a time-share environment. The peripherals used are an inexpensive plotter for design verification and a Gerber drafting machine to produce the artwork. However, the SCAD.PCB data files have been structured to permit interfacing with more sophisticated interactive graphics systems as these become available.

The different facets of a board design are:

(a) the maintenance of a data base of electronic components and pcb blanks;

(b) the selection of the required components from the data base and their placement on the pcb blank;

(c) the encoding of the circuit for the pcb;

(d) automatic assignment of correct pad and drill sizes to component terminals/pins;

(e) link or wire list determination from the encoded circuit;

(f) routing of the link list to arrive at the wire layout;

(g) generation of artwork, production documentation and numerically controlled drill tapes; and

(h) archiving.

The production of a pcb layout using SCAD.PCB is divided into two phases. In the first phase all of the electronic components are declared, their positions on the board defined, and the circuit is encoded. The second phase is the process of finding a wire layout in which a link list of proposed connections between components is defined and then used to steer the router routines. These routines take each link in turn and attempt to find a corresponding wire-path.

SCAD.PCB is a large software system with more than 19 different programs having a combined total of over 20,000 lines of source code. The sequence of operations involving all programs are shown in the Operator Flowchart (Fig. 1). Most of the significant operations are described in the body of this report and more detailed information is contained in the Appendices. Appendix A summarises the hardware and software requirements of the system. Appendix B describes how the data base is organised, and Appendix C the methods of encoding the circuit. Appendix D gives a brief description of each of the programs in the system.

The large size of SCAD.PCB precludes a detailed description at operational level in this report. Assistance at this level may be found in a set of macro-programs [2] (see also Section 5.4 below), written especially to demonstrate a typical pcb design.

2. PHASE ONE: SELECTION OF COMPONENTS, PLACEMENT AND CIRCUIT ENCODING

Phase one starts with data initialisation, that is, all the data for defining the circuit board are entered into the computer. Then follows an iterative process of checking and amending this data until a satisfactory layout of components on the board is obtained, and there are no errors in the encoded circuit.

2.1 Selection of Components

A component is declared to be a part of the circuit board when its name is typed in at the computer terminal. A component specification is completed when the package type for the component and its position are also entered. The package type is drawn from a library of standard package-outlines, and the position chosen for it depends on the user's proposed layout.

2.2 Placement

Placement is the process of finding a position on the board for every component. Currently, the process is a manual one, where the operator determines the coordinates of components for the best pcb layout. Drawings of the components in position on the board are obtained, as in Fig. 2, to assist in the inspection for errors. Such an inspection may detect wrong package type references or incorrect placement specifications. Of particular importance at this stage is the need to detect incorrect package types as these will affect both the placement and the subsequent encoding of the circuit.

Ideally, the placement process should be computer-assisted, but for this to be feasible the circuit must be encoded first, to establish the required relationships between components. The inspection for package reference errors at this stage is important and should be carried out with components placed in temporary positions if necessary.

2.3 Circuit Encoding

A dictionary definition of the word "node" is, "knob on root or branch; point at which leaves spring from stem." This is an accurate portrayal of the situation on a printed circuit board where a component terminal or pin is soldered to a copper pad (knob), which in turn has printed wire-paths (branches) springing from it to go to other pads (knobs). But in circuit analysis a different meaning is assumed. There, a node is taken to mean a collection of such knobs, there being exactly one node for every electrically common point in the circuit. SCAD.PCB adopts the latter definition of a node [3] and the 'knobs' are instead called terminals/pins or vertices, depending on the context in which they are used. The branches are also called links and wire-paths.

When the circuit is being encoded each declared component is treated as a separate entity with a hardware package having a discrete number of pins. In this context the 'knobs' are called terminals/pins. Fig. 3 illustrates how a circuit diagram may be prepared as an aid to circuit data entry at the computer keyboard. Unique node numbers are given to all parts of the circuit that are electrically common and the pins on each component are clearly marked with a pin number. The routine for encoding the circuit systematically introduces each component in turn for the user to assign node

numbers to its pins. All pins with the same node numbers will ultimately be linked together by a network of printed wire-paths.

Computer listings of the encoded circuit enable the user to cross-check his data with the original circuit and note any errors for subsequent correction. If thorough circuit validation procedures are adopted at this stage, before any wire-paths are found, an error free pcb is predicted.

3. PHASE TWO: WIRE PATH LAYOUT

The objective of phase two is to link the pins of each node in a network of printed wire-paths. In this context the pins of section 2.3 are called vertices. The *modus operandi* of phase two dictates that all the vertices of a node be identified and easily accessed. To this end SCAD.PCB creates an alternative listing of the encoded circuit, called the vertex list, in which each node is a set of vertices or pins.

3.1 Definition of the Terms Link and Mask

A link is a specification for a wire-path of given size to be found between two nominated vertices. The term "link" is synonomous with the term "branch" used above in Section 2.3. The name "link" has been adopted because whole chains of vertices are often linked together in a continuous sequence. For each node, a "tree" of links must be brought into being, to cause all the vertices in it to be connected together by a network of wire-paths. After link "trees" have been generated for every node they are all deposited in a single data file called the link list, which is used as an instruction list to direct the automatic routers in their search for wire-paths.

It should be clearly understood that, at this stage, a link is a specification for, and not an actual wire-path. As such it contains a track type-number to specify path-size, a start vertex, and a finish vertex. In the link list both vertices are automatically listed in component name and pin number format, e.g. "IC12" 19. Reduced to their simplest form, vertices are represented by the grid coordinates of their positions on the board. The simplest interpretation of a link is, therefore, a directive for a wire-path of specified size to be found between a start position and a finish position.

Router routines find wire-paths for links and append them to a data file. Within the data file each wire-path is tagged with the name and node number of the link that gave birth to its existence.

A wire-path is a concatenated sequence of path segments which connect the start and finish positions of a link. Each path segment is characterised by a track type-number, the coordinates of the start position of the segment and a direction indicator. The finish position coincides with the start position of the next segment. The track type-number is inherited from

the track type-number of the link, although individual segments may have hand selected type-numbers. The coordinates are in units that reflect physical locations on the board. The direction indicator assumes values in the range 0 to 7 as illustrated in Fig. 4. Direction 7, in particular, indicates that the segment is in fact the terminus of the wire-path. Direction 0 is used when a path segment has a direction other than orthogonal. Track type-number 0 specifies that no printed path be produced for the segment.

Much of the flexibility inherent in SCAD.PCB hinges on the concept of "masking". Masks are simply templates of, and for, wire-paths. A mask is extracted from a wire-path by subtracting the start position coordinates of the path from all segments in the path, so that the coordinates of the start position become $(0,0)$. Conversley, a wire-path may be assembled from a link and a mask by adding the coordinates of the start position of the link to every path segment in the mask. A mask is said to no longer "fit" a link when the end point of the wire path it produces fails to map on to the end point of the link.

A set of masks is extracted from the data file of wire-paths produced by the router routines. At the same time, the links have their track type-numbers substituted (or masked, covered, or hidden from view) with index numbers of masks extracted from their wire-paths. Subsequent referrals to the mask/track type-numbers of links will indicate whether or not wire-paths have been found for them.

A mask may be associated with one or more links, so that any changes to a mask will be reflected automatically in all the wire-paths produced by it. The application of the mask concept enables immediate identification of the links that need rerouting, because after moving groups of components about on the pcb to achieve a better layout, certain components will have moved relative to each other and the links between them will no longer have masks that fit. If a group of components is moved *en block* to a new location on the pcb the links between them will remain intact because no internal relative movement between components has occurred. On the other hand, any links from that group to the remainder of the pcb will sever their mask associations. When a mask no longer fits a link the mask type-number of the link is replaced with the track type-number of the first sequent on the mask being displaced.

In addition to the link-mask described above, two other types of masks have been defined in SCAD.PCB. These are the area-mask for producing ground planes on a pcb and the finger-mask for edge-connectors. The method of handling of all mask types is exactly the same, except that each finger-mask or area-mask is assigned to a dummy link with identical start and finish vertices.

3.2 Link Generation

A tree of links must be generated for every node. A special manual tree growing routine is available when tight control needs to be exercised over the way a tree is cast. This routine is usually applied to those nodes

with relatively large numbers of vertices, such as the ground and power supply nodes. An automatic minimum spanning tree algorithm is used for the remaining nodes where the wire-path layout is not so critical. In either case, individual links can be added to a tree, or modifications can be made to a tree after it has been constructed.

When manual routines are used to modify the structure of a link tree the user may fail to include all the vertices of a node in the tree, or he can inadvertantly fragment a tree into several segments by omitting or corrupting links. In such cases the user's confidence in the integrity of a tree is maintained by an automatic checking routine that reports all inconsistancies [4].

While the trees are being constructed all the links generated are grouped under their respective nodes, but later they are brought together in a single link list, in preparation for routing. An automatic routine is used to check that all links have, in fact, been included in this list.

There are special facilities for rearranging all the links on this list into any order that might improve the number and quality of the wire-paths found for them by the automatic routers. Many schemes for increasing the connection rate by ordering the link list have been canvassed in the literature [5,6], but in the experience of SCAD.PCB they have yielded only a marginal improvement in performance because the connection rate is most influenced by wiring density. This is confirmed in more recent literature [7], where it is stated that once the critical wiring density is reached, the probability of finding wire-paths for links rapidly drops from near unity to a value approaching zero. The main benefit of most ordering schemes appears to be aesthetic appeal and the recommended approach for ordering the link list is the mixture of methods to be outlined in Section 3.4.

3.3 Routing

Routing is the process of finding wire-paths for links. This is a process where most of the routing is done by automatic routines before using manual routines to route the remaining links. The ideal situation occurs when the automatic routines route 100% of the links leaving none for manual routing; when the wiring density is low enough the probability of this occuring is very high.

There are two types of automatic routers used in SCAD.PCB. They both initiate their grids with the wire-paths of pre-routed links and then attempt to route the remainder. As new wire-paths are found they are appended to a temporary data file. The routers also have log files to record the success of failure of each attempt to find a wire-path for a link. A common data base for the routers allows each of them to continue the search for wire-paths from where the other left off.

Most of the paths on a board can be classified into a few basic shapes and the first type of router is a heuristic router, designed to search only for wire-paths of the basic shapes. Because of the small number of path

options available, these routers tend to be much faster than those of the second type, the more conventional Lee Algorithm [8,9] routers. Although this second type of router is much slower, it will find a wire-path for a link if it exists.

After the routers have finished their work, the information appended to temporary data files is translated to the link list and mask file. When converted to the mask file format, the manual routine can be used to modify selected wire-paths and improve the wire layout. Generally, the unrouted links that remain after automatic routing will stay unconnected until the manual routine is used to re-route certain critical wire-paths that are blocking their way. SCAD.PCB allows a user to route critical wire-paths before the automatic routers are called, by creating masks for, and then assigning them to links.

The manual router routine creates and modifies wire-paths indirectly through the changes it makes to masks. In the environment of the mask there is no grid containing pre-routed wire-paths in which a new path can be embedded to check whether it overlaps other paths. Consequently, it is possible to intentionally, and unintentionally, make wire-paths intersect other paths when they shouldn't. This may be acceptable in the short term while the manual routing is in a state of flux, but such intersections must eventually be removed. Visual inspection cannot be relied upon to detect all intersections, so an automaton is needed to do the work. The routine provided for the task reports all overlap violations as it systematically embeds all wire-paths in a high resolution grid.

During the routing process no data is lost as the user switches between the automatic and manual routines and vice-versa. The user can, if he wants, route the board one node at a time until the wire layout is complete, in which case the data in the link list and mask file accumulates as the board design proceeds.

3.4 The Recommended Approach for Routing a PCB

The most successful way to route a pcb is in stages. All node numbers with large numbers of vertices should be routed in a first stage before any attempt is made to route the remainder. The rationale behind this is that the large networks (ground and power rails, and bus nodes in digital circuits) require well defined layouts, which are best achieved while the board is uncluttered with the tree structures of the lower order nodes.

After the nodes requiring a well-defined layout have been routed, all of the links for the remaining nodes can be generated, placed on the link list, and routed *en block*. While on the link list the links are sorted into the order in which they will be routed. Before any sorting takes place each link is given a weight to determine its routing priority. One of the functions used to calculate weights counts the vertices found within the rectangle formed by the intersections of the coordinates of the end points of a link. This estimates the difficulty likely to be encountered by a router in the locale of a link. Difficult links, those with a large number of vertices,

are placed towards the end of the link list. Of the different weight functions tried, this one seems to produce a higher connection rate than the others, and a more appealing wire layout.

Ground and power supply node links need large path-sizes because of the extra current they must carry. For these the routers use grids of low resolution (0.1 inch per grid increment). The other nodes use smaller tracks for their links and may be routed on a succession of grids of increasing resolution (from 0.100 up to 0.0167 inch per grid increment) to improve the connection rate and reduce the overall routing time. This is discussed further in Sections 5.2 and 5.3 below.

4. INTERACTION BETWEEN PHASES ONE AND TWO

Often in the course of phase two it becomes apparent that unnecessary congestion has been introduced into the wire layout by the misplacement of certain components. The system permits a return to phase one for the relocation of such components, without losing all of the wire-paths found in the routing phase. All the links remain intact, but some of those with masks assigned may find them detached because they no longer fit. This only happens between components that have suffered relative movement. If the shift was a minor one no change in the link tree structure may be necessary, but if it was a major shift from one side of the board to the other, the tree structure may need radical revision. Each case should be examined and rectified as the user sees fit.

5. SPECIAL FEATURES

The DEC System 10/20 ALGOL 60 computer language, in which SCAD.PCB is written, has the standard 'own' variables and arrays of ALGOL 60, plus a powerful set of string handling facilities to manipulate variable length strings with different byte sizes. These features are utilised in special data-structure arrays which have variable dimensions, and they contribute significantly to efficient use of computer memory. Computer languages without these facilities place restrictions on maximum board size, numbers of components, through-holes, links, etc., and use more computer memory than necessary.

5.1 The Data Base

A large data-base of different package-outline configurations is maintained by the system. It is divided into two parts, one for pcb blanks and the other for electronic component outlines. These contain information required for the production of realistic drawings of components in position on a board, plus the position and diameter of all pins for drilling holes and

selecting pad sizes. A catalogue of electronic components is also kept. In it, electronic components are classified by description and part number, and have cross-references to one or more of the package-outlines in the basic library. A standard package-outline can be selected for a component, either directly from the basic library, or indirectly through the catalogue library. When selected directly, it is the user's responsibility to ensure that he has the correct package-outline; indirectly, the correct package-outline is automatically guaranteed after a catalogue item is chosen.

5.2 A Scheme of Wire-Path and Terminal-Pad Sizes

Embodied in the system are certain mechanisms for handling different "schemes" of tracks and pads. A standard scheme is generally adopted for most pcb designs but there are facilities for creating a large number of "schemes". The concept behind a "scheme" of tracks and pads plays an important part in the efficiency of SCAD.PCB because when the normal practice of routing a pcb with a single grid resolution is compared with the use of multiple grid resolutions in SCAD.PCB, there is a significant increase in the number of wire-paths found for links accompanied by a drastic reduction in the overall routing time. This section outlines the basic features of the "scheme" concept that help to produce these savings.

Four parameters are needed to completely define a scheme of tracks and pads. The first is the minimum grid increment size (G), or highest resolution grid that can be used by the routers. One of three available resolutions may be chosen by selecting a "scheme-number" (Scheme #). The resolutions are 0.0100, 0.0125, and 0.0167 (10.0, 12.5 and 16.7 mil) for Scheme #1, #2, and #3 respectively. Note that all the tracks and pads in a "scheme" have a domain (Fig. 6a) equal to an integral number of minimum grid increments, that they may fit neatly into the grid of a router.

The second and third parameters are related. The second parameter is the minimum path size (P). The third parameter, which is the clearance (C) between adjacent pads and/or paths on a pcb, is chosen through the relation $C = N_0G-P$, where N_0 is an integer such that C is sufficiently greater than zero. the actual size of a track or pad in a scheme is found by subtracting the clearance from its domain, that is, $P_n = nG-C$ where $n = N_0, N_0+1, N_0+2, \ldots$, up to the maximum size allowed for an aperture on the Gerber drafting machine. Thus the domain of a track or pad on a grid comprises a path (P) with a clearance (C/2) on either side as in Fig. 6a. When two tracks are placed side by side a total clearance of C is obtained between the copper wire-paths (Fig. 6c).

The fourth and final parameter needed to define a scheme is the pad annulus. It is the distance between the outer extremity of the copper of a pad and the hole in its center. This effectively sets maximum hole size of a pad and an upper limit on the pin size that a terminal pad can receive.

The standard scheme of tracks and pads is based on Scheme #3 (16.7 mil minimum grid increment size). The minimum path size of 15.0 mil was recommended by a manufacturer of pcb's. When $N_0 = 2$ the clearance becomes

18.3 mil which is very close to the recommended 20.0 mil for low voltage circuits. The pad annulus of 10.0 mil is small enough to accommodate a useful range of component pin sizes in every pad without compromising the amount of solder interface area between pad and pin. The different tracks and pads that are available in the standard scheme are tabulated in Table I. The track and pad type-numbers in Table I are used for selecting the wire-path and pad sizes. They also correspond to the different work stations found on the aperture wheel of the Gerber drafting machine.

In the remaining part of this section a description is given of the way the routers use the different tracks and pads of a "scheme" for routing a pcb. Particular attention will be given to the way the routers handle different grid resolutions. There are two aspects considered in the following discussion, one of searching for, and another of storing a wire-path.

The simplest modes that a router can use to find a wire-path occur when the grid resolution is a maximum (16.7 mil) and the track size is a minimum (15.0 mil). The search for a path proceeds in the direction of the arrows as illustrated by Fig. 6b examining only one point at a time to see if a path can be found. As the search front (arrow) moves forward the router tacitly accepts that the space within one grid increment on either side of the grid point is a legitimate domain for the wire-path. Then as the wire-path is being stored in the grid both the center grid point (arrow) and the grid points on either side of it are tagged as shown in Fig. 6c.

The concept of searching for a path with a single point front applies at all levels of grid resolution. If the size of a proposed wire-path ever violates its legitimate domain of twice the current grid resolution then the track specification on the link to be searched is adjusted to a smaller path size. When the size of a track has a domain less than the legitimate domain no adjustments are needed. If the routers did not automatically adjust path sizes, but opted for a search involving a front of multiple grid points to cover the increased domain, then they would become unnecessarily complicated and slow because of the increased number of grid points that need to be examined. It is much simpler to reduce the path size.

Storage of wire-paths within the router's grid can be accomplished at all levels of grid resolution and with every path size by the method described above. This method, however, seems to be wasteful of space because on a low resolution grid many of the available tracks can have domains that are less than the maximum domain. For example, when the domain of a track is equal to (or less than) the current grid resolution then it is equal to (or less than) half the maximum domain allowed for the grid. It should not be hard to see that such paths could lie on adjacent co-ordinates, one grid increment apart, instead of the normal two grid increments apart as in Fig. 6c. For this a second storage mode of one grid point can be used. Although the second mode of storage may introduce certain economies of space it precludes the utilisation of larger tracks because their domains would intrude upon the domains of smaller tracks.

With the above arrangements for path storage, wire-paths can be found on successive grids of increasing resolution without interfering with or

overlapping each other. For successive grids of decreasing resolution the same is true, but the restrictions on path-size will apply. The three available minimum grid increments are all fractions of the customary 100.0 mil spacing between terminals on electronic components, i.e. 100/10 = 10.0, 100/8 = 12.5, and 100/6 = 16.7 mil. For the standard "scheme" based on Scheme #3 the successive grids of increasing resolution are 100/1, 100/2, 100/3 and 100/6, or 100/1, 100/3, 100/2 and 100/6.

5.3 Ground Planes or Area Masks

Ground planes play an important part in pcb design. In SCAD.PCB they are easily created by first specifying an outer-loop or boundary for the plane. Inner-loops contained within the outer-loop are zones from which the plane is excluded. The area-mask routine automatically establishes an exclusion-zone or inner-loop around every terminal pad within the outer loop that does not have the same node-number as the area-mask link. The area between the outer-loop and inner-loops is filled in by a routine that traces out a sequence of wire-paths that transverse every grid point.

5.4 Storage and Execution Time Economies

The router programs work from an internal grid model of a printed circuit board. Using the string handling facilities in conjunction with data compression techniques, the total amount of computer memory required to model a pcb has been kept to a minimum.

The automatic routers use the "scheme" described in the previous section to save time and grid storage space. These can be set up to route links using any one of a number of grid resolutions, all of which are a multiple of the basic highest grid resolution. By attempting to route links on a grid with a low resolution first, followed by grids of increasing resolution, instead of a single high resolution grid, the routing time is halved and the connection rate of wire-paths found for links is increased by over 15%. This is because grids of low resolution have fewer grid points and are proportionately faster to search. Since most of the links are routed on the lower resolution grids the overall routine time is drastically reduced. Also, adjacent paths found on a low resolution grid will, in general, have enough space between them to allow the passage of other paths on a high resolution grid thus effectively by-passing the blockages normally caused by links routed at an earlier stage.

5.5 Macro Interpreted Command Programs to Control SCAD.PCB

A Macro Interpreted Command Program is one that will automatically execute a sequence of commands as if a user were engaged in a session at a computer terminal. Such programs are constructed by users to get the computer to automatically carry out often repeated sequences of instructions. These programs can automatically enter and substitute data, and then halt at predefined points to allow the user to take critical decisions before continuing.

Several such programs have been implemented [2] to reproduce the sequences of instructions most often used in SCAD.PCB. One such program called DEMO is used to demonstrate a typical pcb design to potential users. Interspersed with numerous comments and illustrations of corrective modes, the whole demonstration takes about an hour on a computer terminal with a total computer run time of about 10 minutes. In the demonstration all wire-paths are connected without user intervention. Some of the drawings from the demonstration program have been used to illustrate this report.

6. CONCLUSION

The system has been used to design both analogue and logic pcbs involving up to 80 integrated circuits and 1300 wire-paths. Software development has reached a 'user-hardened' stage where mistakes introduced through typing errors etc., are detected before they seriously affect a pcb design. Also, a considerable amount of work has been done to break down the 'man-machine communication barrier' which can, through the use of tortuous data entry procedures and confusing errors messages, inhibit a user's work flow.

SCAD.PCB has shown itself capable of maintaining data integrity by faithfully reproducing the encoded circuit on the pcb without errors. It has proved itself to be flexible enought to handle modifications and special requirements. With its expandable data base, SCAD.PCB avoids the problems that can occur with fixed data base systems where programs often need to be recompiled with extra data storage in order to finish a pcb design.

7. REFERENCES

1. Anon.: "", DEC System 10/20 ALGOL Programmer's Guide (1977), Digital Equipment Corporation, AA-01960-TK.

2. Pietsch, T.D.: "Macro Interpreted Command Programs to Control SCAD.PCB", (May 1981), available from Materials Research Laboratories.

3. Rose, N.A. and Oldfield, J.V.: "Printed-Wiring-Board Layout by Computer", Electronics and Power (October 1971), pp. 376-379.

4. Deo, N.: "Chapter 11 Graph-Theoretic Algorithms and Computer Programs", Graph Theory with Applications to Engineering and Computer Science, Prentice Hall (1974), pp. 258-327.

5. Hightower, D.W.: "The Interconnection Problem: A Tutorial", Computer (April 1974), pp. 18-32.

6. Breuer, M. Ed., et al: "Chapter 6. Routing", Design Automation of Digital Systems, Prentice Hall (1972), Vol. 1, pp. 283-333.

7. Agrawal, P., Breuer, M.A.: "A Probabilistic Model for the Analysis of the Routing Process for Circuits", Network (1980), Vol. 10, pp. 111-127.

8. Lee, C.Y.: "An Algorithm for Path Connections and its Applications", IRE Transactions on Electronic Computers (September 1961), pp. 346-365.

9. Hoel, J.H.: "Some Variations of Lee's Algorithm", IEEE Transactions on Computers (January 1976), Vol. C-25, No. 1, pp. 19-24.

10. Breuer, M.A. Ed., et al: "Chapter 5. Placement Techniques", Design Automation of Digital Systems, Prentice Hall (1972), Vol. 1, pp. 213-282.

11. Hanan, M. and Kurtzberg, J.M.: "A Review of the Placement and Quadratic Assignment Problems", SIAM Review (April 1972), Vol. 14, No. 2, pp. 324-342.

12. Bentley, J.L. and Ottmann, T.A.: 'Algorithms for Reporting and Counting Geometric Intersections", IEEE Transactions on Computers (September 1979), Vol. C-28, No. 9, pp. 643-647.

13. Hosking, K.H.: "A New Technique for the Placement of 'Solder Blobs' and Hence Discrete Components", The Marconi Review (Fourth Quarter 1975), pp. 169-183.

TABLE I

The set of apertures using Scheme #3, based on a minimum grid increment size of 0.100/6 = 0.00167 inch = 16.7 mil
Other schemes available (without apertures) are:
Scheme #1: Grid sizes: = 10.0, 20.0, 50.0, 100.0 mil
Scheme #2: Grid sizes: = 12.5, 25.0, 50.0, 100.0 mil

SUMMARY OF TRACK AND PAD TYPES FOR SCHEME # 3

Grid Sizes: = 16.7, 33.3, 50.0, 100.0
Minimum Track Size: = 15.0 mil
Minimum Pad Annulus: = 10.0 mil
Minimum Clearance: 18.3 mil
Note: Type # 1 is used exclusively for delineating PCB borders (or outlines) and/or lettering.

Type	Track DX	x	DY		Pad DX	x	DY	Hole Size	Lead Size
2.	15.0	x	15.0						
3.	31.7	x	31.7						
4.	48.3	x	48.3						
5.	81.7	x	81.7						
6.	115.0	x	115.0						
7.	148.3	x	148.3						
8.	181.7	x	181.7						
9.	33.3	x	33.3	=	Square for AREA.MASK filling				
11.					31.7	x	31.7	11.7	9.6
12.					48.3	x	48.3	28.3	26.2
13.					81.7	x	81.7	61.7	59.6
14.					115.0	x	115.0	95.0	91.9
15.					148.3	x	148.3	128.3	125.2
16.					181.7	x	181.7	161.7	158.6
17.					48.3	x	81.7	28.3	26.2
18.					81.7	x	48.3	28.3	26.2

APPENDIX A

HARDWARE AND SOFTWARE CHARACTERISTICS

A.1 Computer Peripherals

Only a minimum number of peripherals are required to operate SCAD.PCB and these will be available at nearly all computer installations. They are teletype or simple visual display unit, a line-printer and a plotter. With these a user is able to complete and inspect a pcb design before sending it to a Gerber drafting machine for the automatic production of artwork.

Any type of plotter can be used with the SCAD.PCB programs by compiling and loading a particular plotter software package. If a new software package is needed for a different plotter, then a copy of an existing plotter package can be easily modified. All that needs to be changed are the plotting increment sizes, the maximum physical dimensions over which the unit can operate, and the format of data produced by the PLOT routine to drive the device. Software packages are currently available to drive Zeta Series 100, Tektronix 4006-1, Tektronix 4014 and Hewlett-Packard 7221B machines.

The SCAD.PCB programs automatically use the dimensional information supplied with a plotter package to scale drawings correctly. When a drawing exceeds the physical limits of a plotter the system automatically partitions the drawing into several frames which can be viewed individiually.

A.2 Consideration of Software Limitations

The system has built into it very flexible data storage handling facilities that automatically expand to the physical or logical limits of the computer. This has been made possible through the language in which the system has been written, namely DEC System 10/20 ALGOL 60 [1]. Dynamic memory assignment is used, allowing arrays to expand or contract as a board design progresses, making efficient use of computer memory. SCAD.PCB will, therefore, expand its data arrays to accept unlimited numbers of components, through-holes, links, etc. Other systems with fixed data arrays have limits on these parameters, which, if exceeded cause problems.

The programs asking for the largest amounts of computer memory are the automatic routers, which set up grids containing up to 60 points per inch. For a double-sided 15 x 10 inch board the total number of grid point that must be represented in memory is 1,080,000. If each point were to use just one word of computer memory, the capacity of many a computer would be severely taxed. Through data compression techniques this number has been reduced to about 80 K or 81,920 36-bit words. This is equivalent to 368,640 8-bit bytes. This quantity is a typical maximum value for a board of this size.

In practice, the routers require only a small amount of memory at the start of a job, and as links are searched for wire-paths, the data arrays are progressively expanded to store new wire-paths. The quantity of memory also depends on the number of different nodes a router must handle in a pass through the link list, explained as follows. When a router initiates a grid with the wire-paths of previously routed links, the grid points which represent each path are tagged with a number representing the node of the link. The tags are used by the routers to discern the paths open to each node: paths of the same node may coincide, others may not. A distinct tag is assigned for each node found on the link list with links still to be routed. The paths of nodes that have been completely routed are all given the same tag, because they are no longer considered in the routing process except as constraints on unrouted links. Now the size of the byte of computer memory needed to register a tag on a grid point depends on the maximum tag number, which is equal to the number of unconnected nodes on the link list being routed, plus one extra for all connected nodes and pcb boundaries. The lower the tag number, the smaller the byte size. Thus, a router utilises a minimum amount of computer memory when one node at a time is being added to the link list and routed. A maximum amount of memory is required when all nodes appear unconnected on the link list, but as the links of each node are routed the quantity of memory drops markedly after each pass through the link list, even though new paths are being found and added to the grid.

Multi-layer boards with up to 14 layers can be accommodated. A mix of up to 24 different pad and track sizes can be used in routing a board. This limit is imposed by the 24 photo-plotting apertures available on the Gerber drafting machine, which is used for automatic production of pcb artwork. Link and area masks can be used to create special edge-connectors and ground-planes for a pcb design using the 24 apertures.

A.3 Using the System

SCAD.PCB is set up to be used interactively in a time-share environment. Individual programs are executed as functional entities, to perform specific tasks as and when needed. These programs have names prefixed by PCB, e.g. PCBØA, PCB3A and PCB7, and a guide to their usage is found in the operator flow chart of Fig. 1. This flow chart gives an abbreviated description of the function of each program and a number of decision boxes which define the circumstances under which they should be used.

The flow chart reveals three basic areas of operation; data-base management, component placement, and routing. Programs related to data-base management are PCBØA, PCBØB and PCB2C. Those related to placement are PCB1A, PCB1B, PCB2A, PCB2B and PCB3B. For routing, the programs are PCB3A, PCB3B, PCB4A, PCB4B, PCB4C, PCB5A, PCB5B, PCB5C, PCB5I, PCB6A, PCB6B and PCB7.

A program being executed, will, in general, provide a running commentary on what it is doing, interspersed with requests for instructions and/or data. In many instances a list of available instructions can be obtained by typing the number 99 in response to a * or # at the computer terminal. Default instructions and/or data may be used to minimise keyboard

entry. Wrong responses will evoke error messages that describe the fault and invite the user to try again.

Hatfield Polytechnic Computer Centre has implemented a Macro Interpreted Commands (MIC) program by which frequently used sequences of commands, at monitor and user level, on a DEC System 10 can be called. Macro-programs to execute sequences of SCAD.PCB programs with all the correct responses have been implemented with MIC to simplify the use of SCAD.PCB. They allow a standard pcb design procedure to be adopted, with the option of using individual SCAD.PCB programs to perform specific tasks. Further detail on the macro-programs available, the context in which they are used, and how they are called is contained in reference [2].

APPENDIX B

A DESCRIPTION OF THE DATA BASE

Any electronic circuit has components like resistors, capacitors, integrated circuits, which must be interconnected in a well defined manner by printed wire-paths. This system takes the components and places them on a board in an ordered fashion and then proceeds to work out the printed wire-paths between them.

When positioning the components on a board the main constraints are that they be placed close to each other, but far enough apart to permit the passage of printed wire paths amongst them. For this to happen the physical dimensions of both the pcb blank and the components must be known. To this end, the system has a data base of package-outlines contained in a package-outline library. The user must identify those package-outlines in the library that resemble the components of the circuit.

To assist the user in the selection of a correct package-outline for each component, a stores inventory of electronic parts is kept called the catalogue library. In it, each item, as part of its specification, has a reference to at least one of the package-outlines contained in the package-outline library. Either or both of these libraries may be consulted to select a package-outline, but use of the catalogue library enables the automatic selection of a package-outline from a predefined list, removing uncertainties from the choice. The structure and usage of the libraries are detailed below.

B.1 The Package-Outline Library

This library contains the physical dimensions for over 600 pcb blanks and some 200 electronic components. The pcb blank and component dimensions are held in two computer data files called "BOARDS.LIB" and "PACKGE.LIB" respectively. A computer plot of all package-outlines in these libraries can be obtained where all the items are drawn and numbered in sequence. Each outline is a realistic plan view of the item as shown in Fig. 3. It is not recommended that all the outlines of "BOARDS.LIB" be reproduced since they are large items using much paper to show very little detail, but all the items in "PACKGE.LIB" should be plotted by each user, so that the individual outlines in it can be consulted for selection.

As and when required, new package-outlines can be made and appended to the library, while obsolete outlines may be modified or deleted. See 'New Package Outlines?' of Fig 1. Most resistors, capacitors, etc. have similar shapes but different dimensions. For example, resistors generally have a cylindrical body with a wire protruding from both ends. For these only a few critical dimensions are needed to enable specialised computer routines to create package-outlines. Routines to produce most basic shapes have been implemented, plus a general routine for outlines of any shape.

Examination of a package-outline in Fig. 7 will show that it has five parts. (1) It has a name, e.g. "14-PIN DIP", placed below the outline drawing. (2) There is a position for a circuit name, (IC1) in the place where the library sequence number (Ø15) now appears. (3) The cross-hairs locate the logical center about which the package-outline can be rotated. (4) The position of each pin is designated by a cross (the actual specification requires both a position and diameter for each pin), with the exception of pin number one, which has a cross enclosed in a square. (5) Finally there is the sequence of plotter instructions needed to reproduce the realistic plan view of the package.

A package-outline is selected for a component by using a library sequence number. In the example, IC1=P2, the symbols 'P' and '2' cause the package-outline in position 2 of "PACKGE.LIB" to be selected and assigned to IC1. (The indirect selection of P2 using the catalogue library is covered in B.2.) Thereafter SCAD.PCB keeps tabs on the package-outline for IC1 by using its package name "14-PIN DIP", instead of its library sequency number 2. Should, for example, "14-PIN DIP" now be deleted from "PACKGE.LIB" another package-outline will be found in second position. The automatic crosschecking carried out by SCAD.PCB will now be unable to locate the name "14-PIN DIP" for IC1, and will issue a request for a substitute. If the sequence number had been used as a tag instead of the name, the outline now in second position would have been chosen with possibly disastrous results. The use of names to keep tabs on the outlines assigned to components permits the reordering, insertion and deletion of package-outlines in a library in the knowledge that the integrity of package-outline assignments to components will be maintained.

In addition to the standard package-outline libraries the system supports privately generated and maintained package-outline libraries to supplement the standard libraries. These are "BOARDS.USR" for pcb blanks and "PACKGE.USR" for component packages. New pcb designs very often have a requirement for one or more package-outlines not found in the standard libraries, and to avoid increasing the size of already large standard libraries even further, a pcb's unique package requirements can be kept in the supplementary libraries. A typical selection from a supplementary package-outline library would be IC1=Q2, where the selector symbol 'P' has been replaced by 'Q'.

B.2 The Catalogue Library

The catalogue library is an inventory of electronic components containing about 640 separate items. Location of individual items in the catalogue is facilitated by their classification into groups by function, such as resistors, capacitors, etc. New functional groups may be added to the library, and old groups may be deleted or have their function descriptions altered. New items may be defined for groups and old ones deleted or modified.

Only one parameter is needed to define a new group of items and that consists of a brief description of the function of the group, e.g. TRANSISTOR FIELD EFFECT, or RESISTOR, FIXED, FILM.

Every item within a group is characterised by a Part-Description, Catalogue/Stock-Number, Bin-Number, Quantity-Unit, and one or more package-outline-options. The Part-Description is a brief description of the item such as: MM74CØØ, Quad 2-Input NAND Gate, or 680K Ohm, 5%. The Catalogue/Stock-Number, e.g. 5962-TN-997-3295, is used for accounting and stock control. The Bin-Number, e.g. 159/247, is used by storemen to locate an item. Quantity-Unit is the unit of issue, e.g. 1 or 100, for stores requisition. Each package-outline-option lists the name of a package-outline suitable for use with an item.

The user may select only one of the package-outline-options, but a special feature of SCAD.PCB creates options consisting of chains of package-outline-options. This arrangement causes all of the outlines named in the chain to be merged into a single new package-outline without having to include it in a library. When an option such as this is being defined, each outline named in the chain is also supplied with a 'drawing instruction' for the merging routine to instruct it to include certain parts of an outline and omit others. In this way certain desirable combinations of components can be built up, such as, an integrated circuit mounted in a socket, or a heat sink on a transistor.

All the catalogue library information for component packages is stored in a computer data file called "PCATAL.LIB". A computer listing of the catalogue can be obtained through program PCBØB. A typical selection from the catalogue would be IC1=G1Ø I15 O1, where G1Ø, I15 and O1 specify the group, the item, and the option respectively. Note that the symbols 'G' or 'P' are mutually exclusive, meaning that either the catalogue library or the package-outline library may be used, but not both together.

The standard catalogue libraries are "BCATAL.LIB" and "PCATAL.LIB" for pcb blanks and components respectively. The corresponding supplementary catalogue libraries are "BCATAL.USR" and "PCATAL.USR". Each catalogue library references only one package-outline library, i.e. "BCATAL.LIB" references "BOARDS.LIB", "PCATAL.LIB" references "PACKGE.LIB", "BCATAL.USR" references "BOARDS.USR", and "PCATAL.USR" references "PACKGE.USR". Selections are made from the supplementary catalogue libraries by using the selector symbol 'H' in place of 'G' e.g. IC1=H1Ø I15 O1.

The system has the ability to take all the items selected from the catalogue library and use the information stored therein to create a parts description list of all circuit components (including the extras chosen through the special options) and produce a completed 'materials schedule form' for store requisitions.

The principle of using names as tags instead of sequence numbers, as described in B.1, is also applied to items selected from the catalogue library. Thus, when an item selected for a component is deleted from the catalogue, the cross-checking routines will detect this and request a replacement item, and the user may be sure that the integrity of the assignment of items to components will be maintained.

APPENDIX C

HOW TO ENCODE A CIRCUIT OR DATA PREPARATION

Before a pcb design is started, sets of data must be prepared for entry at a computer terminal. All of the electronic components in the circuit must first be named and have suitable package-outlines selected for them. Then the positions on the board for these package-outlines must be worked out. Finally, to define the circuit, a node number must be assigned to every component terminal (Fig. 3).

In this appendix consideration is given to different aspects of data preparation that will facilitate an orderly approach to the use of SCAD.PCB.

C.1 Circuit Partitioning

When a circuit is large it is often advantageous to partition it into smaller circuits, complete the design of each small circuit, and recombine them later into a single large board. An alternative to recombining sub-circuits is to create a back-plane on which any interconnections between the sub-circuits are made.

The system has built into it the ability to handle up to 99 sub-circuits or sections, but with no facilities, as yet, for automatic circuit-partitioning into sub-circuits (See D.4).

C.2 Tabulating Components and Package-Outlines

Every section is made up of a blank pcb plus a number of electronic components to be mounted on it. A special form (Fig. 8a) has been prepared where a user can tabulate the name, package-outline specification, and position of the package-outline on the board. In this section the method for tabulating component names and outline specifications is described; section C.3 deals with component placement.

Each component in the circuit should be given a unique name starting with the pcb blank itself. This is always the first component in any section, and without it a section cannot exist. There is no limit set on the number of characters in the name of the board. Names of electronic components, by contrast, are only allowed a maximum of 5 characters, usually one or two alpha-characters followed by one or more digits, e.g. R1, R2, IC1, IC16, C3 etc. All alpha-characters must be entered upper-case.

Package outlines are selected from the "BOARDS.LIB" library for the pcb blank and "PACKGE.LIB" for every other component in a section. The principles for selecting package-outlines were discussed in Appendix B.

C.3 Component Placement

The process of working out a location for each package-outline on the blank circuit board is called component placement. This is a difficult task as individual components must be located close to each other without overlapping to minimise the length of wire-paths between them, yet far enough apart to reserve sufficient space for the passage of all wire-paths.

This task is an obvious candidate for the interactive and automatic placement procedures used by PCB3B, but for these to operate the encoded circuit must be available. Methods for encoding circuits are discussed in Appendices C.4 and C.5, and placement methods are here restricted to manual specification. The present discussion also forms a basis for a discussion of automated methods later.

Once a decision on component positions has been reached, the next task is to encode these locations. This is achieved by giving the coordinates of the logical center of the package, expressed in mils relative to the lower left hand corner of the blank pcb, and an integer giving the anticlockwise rotation of the outline from its standard position. Negative coordinates will be rejected by SCAD.PCB. As an example, the coordinates (1.050, 3.500) in inches becomes (1050, 3500) in mils. (The Imperial system of measurement has been adopted because the spacing between the terminals of most electronic components is standardised to fractions or multiples of 0.100"). An outline can take one of four orientations: Ø, 1, 2 and 3 corresponding to Ø, 9Ø, 18Ø and 27Ø degrees rotation in a counter-clockwise direction about the logical center. The package-outline drawings of the libraries are always located with Ø orientation. Thus, a typical placement specification could be X1Ø5Ø Y35ØØ R3. (Note: The efficiency of the automatic router programs is greatly increased by arranging for all component terminals to be on a 0.100 inch or 100 mil grid where possible).

C.4 Encoding a Circuit

Further to the discussion in 2.3 of the basics for encoding a circuit, this section describes advanced features of the system that will give a user greater control over the way a wire layout is produced.

Every electronic component has a number of terminals/pins that are inserted in specially prepared through-holes on the board. Each through-hole consists of vertically aligned terminal-pads on each layer of a multi-layer board with a hole drilled through the center of them. On the inside perimeter of the hole is deposited a layer of copper which connects all the pads together electrically. The final inside diameter of the through-hole must be large enough to take the pin. The system also permits pins with zero diameter, where no electrical interconnection between layers occurs, to reserve special points on a board.

In the context of circuit definition a distinction between the zero diameter pins and the non-zero diameter pins must be made; they will be called dummy-terminals/pins and, (normal) terminals/pins respectively. The

(normal) pins occur in a package-outline exactly as they appear in the manufacturer's specifications, numbered pin for pin in a counter-clockwise direction. Dummy-pins are generally included in a package-outline by SCAD.PCB and have no relation to a manufacturer's specifications at all; the reasons will become apparent in the following paragraphs.

Pertinent to this discussion is a clarification of the way in which the router programs use a grid to model the pcb. As stated in A.2, the grid points that represent a wire-path are assigned tags which represent the node of the path. Further, the points at which the edges of pads and/or tracks intersect (as illustrated in Fig. 6c) and the grid points traversed by the boundaries of the pcb are all assigned the tag of the node **NMAX**. **NMAX** is called the forbidden node and is equal to the maximum node number for the circuit plus one. It has the special property of not being able to generate any node tree of links; as well, any grid points tagged with **NMAX** will not permit the passage of new wire-paths for any node. All the remaining grid points that have not been traversed by boundaries or wire-paths are tagged with the node ∅ (zero), called the free node because grid points tagged with ∅ are free to be traversed by new wire-paths of any node. Like **NMAX**, the free node cannot generate any links. Of course, the grid points of wire-paths of a particular node will allow new wire-paths of the same node to coincide with them.

How are the two types of terminals represented on the grid of the pcb in the light of the model above? For a (normal) pin every layer of a multilayer pcb must assume the node number of the pin. This being the case, the (normal) pin cannot be assigned the free node number because the new wire-paths of any node that are allowed to cross the pin would be short-circuited by the through-hole. All (normal) pins are initiated with the forbidden node, and not the free node, to avoid any possibility of short-circuits occurring between nodes. A dummy-pin is always initiated with the free node because there is no chance of a short occurring between board layers.

In SCAD.PCB there is one mode for encoding (normal) pins and another for encoding dummy-pins. For (normal) pins, only one node number is assigned per pin, and the routers will automatically tag every layer of the board at the position of the pin with the same node. Dummy-pins are assigned one node number per layer.

As the context shifts to that of link creation, both the dummy-pin and the (normal) pin are represented on the vertex list differently. For every (normal) pin there will be exactly one multi-layer-vertex; each dummy-pin will have several uni-layer vertices, one for each non-zero node number layer and one for every free or zero node number. Up to this point the layer specification on every start and finish vertex of a link has not been mentioned, because, with multi-layer-vertices a wire-path can start from or finish on any layer whatever. In practice, start and finish layers are automatically assigned to a link every time a multi-layer-vertex is used, and the routers will treat these specified layers as being preferred but not essential. The wire-path to or from a uni-layer-vertex must be on the predefined layer of the vertex.

How then can dummy-pins be effectively utilised? One way is to declare components with dummy package-outlines having one or more dummy pins. Such components can be placed at strategic positions on the pcb to produce certain desirable effects when it is being routed. For example, a user may require a common ground point for a pcb; a dummy-pin is an ideal way of ensuring that the different parts of a circuit will be linked and then routed to a common point. This idea can be extended by replacing a single rather long link with several intermediate links; dummy-pins are placed at strategic points along the proposed path of the long link to segment it into a link tree of several small links. Also, many of the package-outlines in "PACKGE.LIB" possess dummy-pins in their configuration that, with careful planning, could be utilised without needing to declare dummy components. (The use of dummy pins may be extended at some later stage with the development of a routine, in PCB3B, based on graph theory, to automatically generate and route link trees that are planar).

C.5 Tabulating the Circuit

When encoding the circuit the electronic components of a section are presented to the user in sequence. As each component is presented the node numbers are assigned to its terminals one by one. Forms are available for tabulating the node numbers for the (normal) pins and dummy-pins in Figs. 8b & c.

A useful convention to adopt when assigning node numbers is to reserve the lower order node numbers for those link trees with the greatest number of branches, as this reduces the number of keystrokes on data entry.

APPENDIX D

A DESCRIPTION OF THE SCAD.PCB PROGRAMS

A brief description of each program found in the operator flowchart of Fig. 1 is contained in this appendix to provide more information on different aspects of the system.

D.1 PCBØA: Maintenance of the Package Outline Libraries

When an electronic component is to be incorporated in the catalogue library or directly used in a pcb design and there is no suitable package-outline in "PACKGE.LIB" to associate with it, a new package-outline must be appended to "PACKGE.USR" before a pcb design can proceed. Similarly, a new board-outline may need to be added to "BOARDS.USR" (see Appendix B). Program PCBØA is designed to create new board or package-outlines and insert them in either of the package-outline libraries, as well as perform other maintenance functions, such as deletion and updating.

To avoid confusion in a multi-user environment any access to modify the libraries "BOARDS.LIB" and "PACKGE.LIB" should be limited to a few persons. Board and package-outlines peculiar to a pcb design should be generated by the user in PCBØA and stored in the supplementary libraries "BOARDS.USR" and "PACKGE.USR". As the formats of the data in the standard and supplementary libraries are identical, the manner in which modifications to a library may be carried out is the same in all cases.

After creating a new outline it is advisable to use PCB2C to draw it and then check it before incorporating it in one of the libraries. After its inclusion in a library, PCB2C is again used, this time to produce a drawing of a complete list of outlines in the library to use as a reference when selecting package-outlines for electronic components. This is important, as all the library sequence numbers from the point of insertion of a new outline will have been altered.

D.2 PCBØB: Maintenance of the Catalogue Libraries

The catalogue library is an inventory of the items held in a local store which will need periodic updates. When an update occurs all users will be supplied with a new computer listing of the library. If a pcb design has items which have subsequently been deleted from the catalogue library the automatic cross-checking carried out by SCAD.PCB will uncover these items and request replacements.

Program PCBØB is used specifically for updating the catalogue libraries. New items can be appended, and existing items can be modified or deleted. Many other functions related to these basic functions can be

performed on the library. But, as with the package-outline library, an updated listing of the catalogue library should be used when selecting items, otherwise the wrong component may be chosen when library sequence numbers are incremented or decremented by insertions and deletions.

As items in the catalogue library have cross-reference links to the outlines in the package-outline library, new outlines must be included in the latter library before they can be assigned to new items in the catalogue library. Conversely, items should be deleted from the catalogue first, and a list of package-outlines still used by the catalogue library obtained before deciding which outlines can be deleted from the package libraries. In either case, comprehensive internal checking procedures will inform the user if the cross-linking has been abused. All internal checking is carried out using the 'names' of outlines, not sequence numbers, thus preventing a source of confusion when the order of package-outlines is altered by PCBØA.

When catalogue libraries are created or modified they can only refer to the outlines contained in one package-outline library. Thus, "BCATAL.LIB" refers to "BOARDS.LIB", "PCATAL.LIB" refers to "PACKGE.LIB", "BCATAL.USR" refers to "BOARDS.USR", and "PCATAL.USR" refers to "PACKGE.USR". Note that the use of supplementary libraries, as outlined in D.1, is extended to the supplementary catalogue libraries.

D.3 PCBØC: Generation of Gerber Drafting Machine Apertures

A set of Gerber drafting machine instructions is generated for creating a new set of pad and track aperatures for a specified wiring scheme. The amount of work involved in setting up new apertures on a Gerber drafting machine is considerable and should be avoided unless absolutely necessary. Modern laser machines, which also accept Gerber machine instructions, require much less effort to set up new apertures.

D.4 PCB1A: Component Selection and Placement

This program has several data-entry modes wherein a user enters the component name, package specification and the placement specification of every electronic component in the circuit. The entry of placement data is optional, depending on whether or not the user is going to use automatic placement. The program creates three data storage files on disk which form the data base of the board being designed. The first is "COMPNT.DAT", which stores the name, package-outline name, and placement of every component. The others are "BOARDS.DAT" and "PACKGE.DAT", which are subsets of the package-outlines contained in "BOARDS.LIB" (or "BOARDS.USR") and "PACKGE.LIB" (or "PACKGE.USR") respectively. These sub-files are formed containing only the minimum number of outlines required by "COMPNT.DAT", which is significantly less than the total number available. An auxilary file called "STORES.DAT" is also formed when items are selected from the catalogue library, "PCATAL.LIB" (or "PCATAL.USR"), and this serves as a cross-reference between "COMPNT.DAT" and "PCATAL.LIB" (or "PCATAL.USR"), for producing a parts list and materials schedule.

When a pcb design is to be started, the user must first specify the different sections or sub-circuits (see Appendix C.1) that are to be used. In so doing, the package-outlines selected are pcb blanks from "BOARDS.LIB" (or "BOARDS.USR"). Later, when the components are being specified, the package-outlines are taken from "PACKGE.LIB" (or "PACKGE.USR").

There are instructions for rearranging the order of components within a section. Once a reordering has taken place, the names of components may be changed by reindexing them. To illustrate the process, consider the initial sequence of resistors as an example: R1, R2, R3, R4, R5. After reordering the sequence becomes: R5, R2, R1, R3, R4. On reindexing, R5 becomes R1, R2 is unchanged, R1 becomes R3, R3 becomes R4, and R4 becomes R5. This renaming enables groups of components in close proximity to each other on the pcb to have similar index numbers, to facilitate component identification. As well as changing the order of individual components for reindexing, all components may be sorted, section by section, with the component names in alpha-numeric order.

In PCB1A components with incorrect package-outlines may be corrected and those badly placed may be repositioned. Facilities for shifting individual, groups of, or whole sections of components from one part of a pcb to another whilst being rotated are also available within the program. Interactive placement on a graphics terminal with PCB3B is the preferred alternative, after the circuit is defined with PCB1B.

Instructions for partitioning circuits (see C.1) are available in PCB1A. Sections may be deleted, split or merged. Individual components may be shifted from one section to another or deleted. The sphere of influence of these operations extends into the data files created by subsequent programs. These are "CIRCUT.DAT" handled by PCB1B, the "TERMNL.ϕ??" terminal data of PCB2B, and the linking data of PCB3A and PCB3B, namely "LINKIN.ϕ??", "AMASK.ϕ??" and "LMASK.ϕ??". These data files are automatically updated by PCB1A to reflect all sectional modifications.

The splitting and merging of sections is illustrated by the following example. Say, for example, a group of components in section #2 is broken off and shifted to section #3. In "CIRCUT.DAT" the group will be transferred from #2 to #3. The "TERMNL.ϕϕ2" file will have all data relating to the group copied to "TERMNL.ϕϕ3" and then deleted from it. Similarly, all the relevant data will be extracted from "LINKIN.ϕϕ2", "AMASK.ϕϕ2" and "LMASK.ϕϕ2" and transferred to "LINKIN.ϕϕ3", "AMASK.ϕϕ3" and "LMASK.ϕϕ3". However, the linking data may contain links between components in the two sections, which now belong exclusively to neither section. This data is diverted to the data pool of "LINKIN.ϕϕϕ", "AMASK.ϕϕϕ" and "LMASK.ϕϕϕ", where it can be either deleted or saved until required for merging the two sections.

D.5 PCB1B: Entry and Amendment of Circuit Data

Before any wire-paths can be found for a pcb the encoded circuit data must be entered at the computer terminal and stored in a data file called "CIRCUT.DAT" using program PCB1B. A listing of "CIRCUT.DAT" can be obtained

for cross-checking the stored data with the original circuit, and if any errors are found or modifications are necessary, PCB1B is again used to introduce these changes.

Initially, PCB1B sets up data arrays to receive a circuit according to the package-outline information stored in "COMPNT.DAT". For each component package a data array of pins is initiated; (normal) pins are preset to the forbidden node NMAX and the dummy-pins are preset to zero. PCB1B then checks the existence of "CIRCUT.DAT", and if it is available, the old encoded circuit is read into the data arrays, replacing the preset nodes with the nodes of the encoded circuit. As this process is taking place, checks are carried out to ensure that the data from "CIRCUT.DAT" can still be properly placed in the data arrays. If not, the preset nodes are not replaced by the nodes of the encoded circuit, and the user must reassign nodes to the components affected in this way. If "CIRCUT.DAT" is unavailable the preset pins stand as they were initiated. PCB1B has now initiated its data arrays and is ready to encode a complete circuit or change an existing circuit. To do this the user must call the apprioriate instructions in PCB1B which enable him to scan the pins of each component and assign node numbers of them. After entering and/or modifying an encoded circuit another instruction can be called to dump the encoded circuit in the data file "CIRCUT.DAT".

D.6 PCB2A: Drawing of Components in Position on the Board

All the data stored in "COMPNT.DAT", "BOARDS.DAT" and "PACKGE.DAT" are relatively meaningless unless they can be transformed into a picture for inspection (Fig. 2). PCB2A takes the above data files and uses them to generate graphics output consisting of a fully labelled drawings of components in position on the pcb. Several options are available for controlling the amount of detail produced in these drawings.

This program may be loaded with one of several available plotter packages (see A.1) which produces data in the format required to drive a specified device. PCB2A automatically partitions the drawing into frames when the scale of the drawing causes the physical limits of the device used to be exceeded.

If some components of a layout have the wrong package-type or have been badly placed the user should return to PCB1A and make the necessary modifications.

D.7 PCB2B: Create and/or Modify Terminal Pad Data

A printed circuit board has large numbers of terminal pads that provide an interface between the pins of components, and the wire paths of the board. Each pad must have a hole large enough to receive the intended pin, with a copper annulus around it that is large enough to satisfy mechanical and electrical interface requirements.

Program PCB2B takes into its data arrays all the relevant information found in the data files called "BOARDS.DAT", "PACKGE.DAT" and "COMPNT.DAT". From these data all pin positions on the blank pcb are computed and placed in arrays with their pin diameters. The user selects a wiring "scheme" (see Table I) which will determine the pad and track (wire-path) sizes that can be used for routing the board. He also selects a set of drill sizes to match the pin sizes of the components, and each pin is allocated a drill size by the program. Then the smallest pad size that produces a minimum standard copper annulus around the drill size assigned to each pin is automatically chosen for each terminal position. Larger pad sizes can be manually assigned in special circumstances.

After the pad and drill assignment is complete, all the accumulated terminal information is dumped into data files called "BORDER.Ø??" and "TERMNL.Ø??", ready for access by other programs. If any changes are introduced to "COMPNT.DAT" that force an update of "CIRCUT.DAT", namely insertions, deletions, renaming or reordering of components, or changes in package-outlines, then a complete reassignment of pads is needed, but this is accomplished very quickly. When the changes to "COMPNT.DAT" are confined to placement specifications, the old "TERMNL.Ø??" data file should be accessed and updated immediately, so as to implement all position changes automatically; the original assignment of pads is not changed.

D.8 PCB2C: Drawing the Package-Outline Libraries

This program is similar to PCB2A and PCB2B in many respects because it uses the same algorithms to draw package-outlines and compute terminal positions. Its main function is to create a drawing of all the package-outlines contained in "BOARDS.LIB", "PACKGE.LIB", "BOARDS.USR", "PACKGE.USR", "BOARDS.DAT", "PACKGE.DAT" or any other temporary library file created for checking purposes by PCBØA (Fig. 7).

Each package-outline in such a data file is drawn with its name included underneath it, with cross-hairs to mark its logical center and a pseudo-component-name (which is its sequence number in the file). All package-outlines are drawn in sequence, to provide a handy cross-reference to use when selecting outlines for assignment to components.

D.9 PCB3A: Generation of Linking Data

Program PCB3A carries out the basic function described in 3.2, i.e. the creation of a tree of links for every node in the circuit. PCB3A uses the node numbers of component terminals contained in "CIRCUT.DAT", as well as their positions as found in "TERMNL.Ø??", to create and/or modify sets of links and store them in the link list data file called "LINKIN.Ø??". PCB3A also creates and/or modifies different types of masks and stores them in data files called "AMASK.Ø??" and "LMASK.Ø??".

In this program the handling of the link list is subdivided into four distinct segments. These are: (a) input of old link list and mask

data; (b) generation of new links and masks and/or modification of old links and masks; (c) the ordering of the link list; (d) output instructions for creating and/or updating the link list and mask data files.

Before any of the above operations can be carried out the program must initiate its own data arrays. On entry to the program the circuit and terminal data are assimilated into a data structure that associates every terminal in the circuit with a physical position on the board, in the form of a vertex list. The minimum number of links needed to form each node tree is known from the numbers of vertices in the vertex list, and empty data arrays are set up to receive links as they are created. These data arrays are automatically extended when superfluous links are generated for special purposes.

D.9.1 PCB3A: Input of Old Linking Data

When a pcb design is in progress, the accumulated data stored in the link list, "LINKIN.Ø??", must be read into the empty link data arrays before any amendments can be made or extra links appended to them. Facilities also exist for reading data from the different mask data files.

D.9.2 PCB3A: Generation of Links and/or Masks

Various options allow a user to create a tree of links for each node by automatic or manual methods, and initiate changes to individual links within a tree. Masks can also be created and/or modified and assigned to links in this segment of the program. Entire node trees or portions thereof can be referenced at the user's computer terminal. Masks can be referenced in like manner, where a cross-reference list of the links to which a mask has been assigned appears, followed by a segment by segment listing of the mask itself (Fig. 4). Finally, there are checking procedures to ensure that no node tree is fragmented or remains unconnected, and other checking procedures to make sure that masks still 'fit' the links to which they have been assigned.

Links for a node are automatically generated using a minimum spanning-tree algorithm [4], which is described below. This algorithm maintains two lists of vertices; the first contains those vertices that have already been connected together in the tree of links; the second list contains those vertices not yet linked to the tree. The first list is initiated with one vertex which acts as a seed. The algorithm examines each vertex combination between the two lists in turn, to find the pair of vertices with the minimum distance before creating a link between them. After this the vertex on the second list is transferred to the first list. This procedure is repeated until no vertices remain on the second list.

Links can be manually generated for a node with the aid of a special tree-growing routine as follows. The user first obtains a map of the vertices belonging to the node by using PCB4C in conjunction with PCB6A, to produce a drawing which shows the physical position of the pads of all vertices on the

board, where the pads of the node for which a tree of links is to be constructed are replaced by vertex numbers, so that they can be identified. The user joins these vertex numbers together by pencil lines to establish a proposed link-tree map. This map is followed as the user invokes the tree-growing routine to create a link-tree.

SCAD.PCB currently supports three different mask types with a reserve capacity for implementing other types. The first is the link-mask described in 3.1, which is a detached wire-path between two points that should be assigned to at least one link. The second is the area-mask which is in effect a ground-plane, where a special routine is used to establish an outer loop (or ground-plane boundary) followed by a number of inner loops around reserved positions within the boundary. All the space between the inner loops and the outer loop is then filled in automatically. Such masks do not have start and finish points to allow them to be assigned to one of the conventional links, so a special dummy link, consisting of a vertex connected to itself is created every time an area mask is assigned. The vertex to which the mask is assigned serves as the reference point for positioning the mask on the board. The third type of mask is the finger-mask, which is used for implementing edge-connectors and the like. An edge-connector is basically an area-mask with dimensions too small to permit the establishment of loops. As such, the finger-mask is generated by the link-mask routine and then assigned to a special dummy link as an area mask is.

D.9.3 PCB3A: Ordering the Link List

In this third segment, all links that have been generated are placed on a link list and sorted into an order specified by the user. The objective of this process is to find an order for the links that is conducive to the router programs finding a maximum number of wire-paths [5,6]. According to recent literature [7], the most significant factor affecting the connection rate is the wiring density of the board. It states that once the wiring density exceeds about 35%, the probability of making a connection drops from a near certainty to virtually zero. Despite this, ordering of the link list has been retained because it often assists in producing an aesthetically pleasing wire layout, and provides a limited increase in the total number of wire-paths found on a densely wired pcb.

Several weight functions are available for the user to assign a weight to each link. Often, large numbers of links will have identical weights, and so, second, third and fourth order weight classes may be added to each link to serve as tie-breakers when the preceding weights are equal. The user may also assign different classes of weights to subsets of the total link list. There is a weight function to give priority to short links; the pin density function of section 3.4; a weight function that gives priority to links on a specified layer; another that gives priority to links with uni-layer vertices; and one to give priority to links bounded by rectangles whose aspect is more that of a square than a rectangle.

D.9.4 PCB3A: Output Instructions

In this fourth segment of the program there are instructions for updating all of the link list and mask data files plus others for obtaining cross-reference lists of vertices, links and masks.

*D.10 PCB3B: Integrated Placement, Linker and Router**

PCB3B gives the user the facility to manipulate images of objects interactively on a graphics display terminal. The component and package data of PCB1A, the circuit data of PCB1B and the terminal pad data of PCB2B are melded to provide spatial information with the required connection relationships between components, to aid the interactive placement process.

Components are initially sorted into one of two sets called two-pin components or cluster-seed components. The two-pin components are then assigned to clusters in tiers, where each tier is a subset of two-pins that can be linked to the seed or preceeding tier of a cluster. Placement first occurs at the macro level where succeeding tiers of two-pins are placed about cluster-seeds, then at a global level where clusters are placed in relation to each other. All placement is constrained by the required connections between components.

Because connection relationships are used in the placement process, these relationships are readily extended to the creation of the links needed for routing. Both the manual link preparation and mask generation for links described in D.9.2 are also subjects of interactive processes on a graphics terminal [10-13].

D.11 PCB4A: Data File Generation for the Router

PCB4A takes the data accumulated by earlier programs and combines it into a data format suitable for the router programs. Information from the data files "CIRCUT.DAT", "BORDER.ϕ??", "TERMNL.ϕ??", LINKIN.ϕ??", "LMASKS.ϕ??" and "AMASKS.ϕ??" are merged to form data files "LINKI.ϕ??", "TERMS.ϕ??", "LINKED.ϕ??" and "SECTON.ϕ??". "LINKI.ϕ??" is a list of the links contained in "LINKIN.ϕ??", where all the vertex names have been replaced by their physical dimensions held in "TERMNL.ϕ??". "TERMS.ϕ??" is a subset of the data in "TERMNL.ϕ??", rearranged into a format suitable for router initialisation. "LINKED.ϕ??" is a record of those links that have had wire-paths or masks found for them. "SECTON.ϕ??" contains all the information needed to initialise the router grids with pcb borders and the wire-paths already found, and is obtained by combining the data found in "BORDER.ϕ??", "TERMNL.ϕ??", "LINKIN.ϕ??", "LMASKS.ϕ??" and "AMASKS.ϕ??", where the vertex, names used to define links are substituted with their physical dimensions.

---

* PCB3B is currently being developed (August 1986).

D.12 PCB4B: Control Files for the Router Programs

The operation of each router program is regulated by a control file ("PCB5A.CTL", "PCB5B.CTL", "PCB5C.CTL", or "PCB5I.CTL") set up through a user dialogue with PCB4B. Each control file consists of one or more control lines, where each line corresponds to a single pass on a router. In each pass the grid resolution on which wire-paths are to be found is set, together with certain parameters that control the routing functions of the router programs.

PCB4B also allows the user to set up a batch control file to run several of the router programs without user intervention, followed by PCB6A to generate graphics display.

D.13 PCB4C: Control Files for Graphics Output

The operation to produce graphics output of the wire-layout is regulated by control files ("PCB6A.CTL", "PCB6B.CTL") that are set up through user dialogue with PCB4C. The user has various options available for him to label wire-paths and draw selected portions of a pcb.

D.14 PCB5A, PCB5B, PCB5C, PCB5I: Router Programs

Router programs find wire-paths for the links listed in "LINKI.ϕ??". Each of the above programs have different functions and capabilities, but they are all initialised in the same way. First a grid is set up of the resolution specified in the control file. Then the terminal pad positions from "TERMS.ϕ??" are marked on the grid using the node numbers assigned to them. After that, all the pcb borders and pre-routed links found in "SECTON.ϕ??" are similarly marked on the grid. Finally, the router works its way through all the unrouted links found in "LINKI.ϕ??", attempting to find paths for them. As each wire-path is found, it is marked on the grid and appended to "SECTON.ϕ??".

When a router program begins execution it sets up several temporary control data files which keep a running record of its status. All of these files are periodically updated, so that if the computer system should 'crash' at any time, all the data accumulated up to the time of the last update will be retained. When the computer system is restored to operational status after a 'crash', the router will automatically use these temporary control files to continue its execution from the point of the last update. When the router finishes execution, it deletes all of the temporary files.

D.14.1 PCB5A: Router Program

This is a fast router which gains its speed through the type of search algorithms it uses, and the fact that only one printed circuit layer is searched at a time. It takes each layer of a board in turn and sets up a grid on which it attempts to find wire-paths for unrouted links. An option exists

for the user to specify whether paths are found either on a designated layer, or on the first available layer. This router is most effective for single layer boards or multi-layer boards without through-holes.

There are three algorithms from which to choose, and either a single or a combination of algorithms can be selected for a pass. The first algorithm will attempt to route only those links whose two end points lie on a straight line or grid coordinate. The second algorithm tries to connect only those links whose end points lie on two disjoint sets of coordinates. Both of these algorithms are restricted in the shape of the paths they may find, and as a consequence may not find a path even though it exists. A third algorithm is included, called the Lee Algorithm [8,9], which has the advantage of being able to find a path of any shape if it exists, but is much slower because it examines all path options. A common method of increasing the speed of this router is to restrict the search area to a rectangle on the grid that surrounds the two end points. If a user elects to use the Lee Algorithm, he will have to specify this boundary. A secondary effect of introducing this boundary is to eliminate those paths that tend to become unacceptably long due to local congestion.

D.14.2 PCB5B: Router Program

Here, the Lee Algorithm is used exclusively in a multi-layer mode, searching on several layers to find a path for a link. It is slower than the algorithms used in PCB5A, and it also uses more computer memory to store the extra layers. But a wire-path is more likely to be found amoung several layers than on a single layer. Similar boundary restrictions to those used in PCB5A are placed on the router to improve its efficiency. The Lee Algorithm can be called with constraints that cause wire-paths with certain characteristics to be chosen in preference to all others. After trying many constraints, the best strategy to emerge was to give priority to placing orthogonal wire-path segments on opposite sides of a two-layer board. This, more than anything else, improves the routability of a board, despite its tendency to produce superfluous through-holes.

D.14.3 PCB5C: Router Program

This also is a fast router, with extended versions of the first and second algorithms of PCB5A. These find wire-paths with orthognal segments on opposite sides of a two-layer board joined by through-holes. PCB5C and PCB5B are designed to complement each other. A single or multiple passes through PCB5C should be used to find over 90% of the wire-paths, followed by several passes through PCB5B with successively higher resolution grids to route any remaining links.

D.14.4 PCB5I: Automatic Wire-Path Inspection

Often, the creation and modification of wire-paths through the mask routines described for PCB3A can leave intersecting wire-paths. Visual inspection is not sufficient to detect all inconsistancies. This program is

used to certify that the final wire-layout contains no errors, or if it has errors, to report where they have occurred on the pcb.

In common with the router programs a grid is set up, in this case the highest possible grid resolution is used. Then the pads from "TERMS.ϕ??" are embedded in the grid, followed by the pre-routed links found in "SECTON.ϕ??". It makes no attempt to route any links, but as each pad and wire-path is being embedded a check is carried out to ensure that the node of the link segment being inserted in the grid does not clash with a different node number already in the grid. If it does, the clashing segment only is appended to "SECTOI.ϕ??", as a wire-path.

D.15 PCB6A: Wire Layout for Inspection

Output data files "TERMS.ϕ??", "LINKED.ϕ??", "SECTON.ϕ??" and "SECTOI.ϕ??" from PCB4A, PCB5A, PCB5B, PCB5C and PCB5I, are used to draw the current wire layout for inspection (Fig. 5). Various options for controlling the type and amount of output from PCB6A are available through the control file "PCB6A.CTL" from PCB4C. These include overlays for multilayer boards, frame selection to view only important parts of a board, and facilities to draw different combinations of connected and unconnected links.

D.16 PCB6B: Driving Instruction for the Gerber Drafting Machine

All the data files used by PCB6A with the exception of "SECTOI.ϕ??" and including "TERMNL.ϕ??", are accessed by PCB6B to generate instructions for a Gerber drafting machine to produce the pcb artwork, drilling instructions, and a summary of required apertures and through-hole sizes.

D.17 PCB7: Post-Router Link Mask Extractor

All of the wire-paths contained in "SECTON.ϕ??" are examined by PCB7 to see if any new link-masks can be extracted from those links recently connected by the routers and appended to "LMASKS.ϕ??". If there are, the track-type numbers contained in the link specifications of "LINKIN.ϕ??" are updated by overwriting them with the indices of the link-masks of their wire paths. The user can then modify wire-paths found by the routers by changing their masks in PCB3A.

D.18 PCB8: Archiving

All the data files that have been generated thus far can be combined into one large data file ready for the user's archives. If, in the future, the board design needs to be updated, the data file can be decomposed or broken down into its constituent files again, ready for access by SCAD.PCB. The archiving facility permits a user to retain backup versions of a job and have more than one job in progress at any-time.

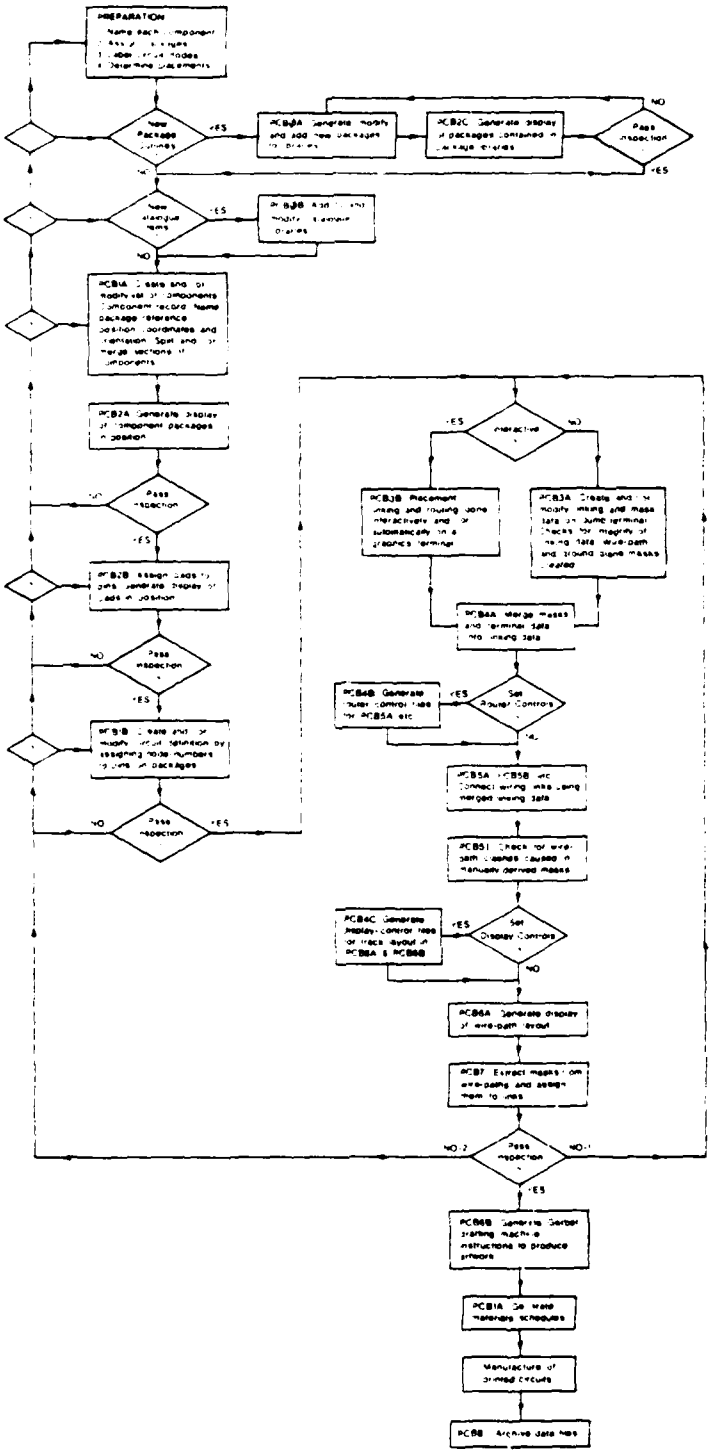


FIG. 1 Operator Flowchart.

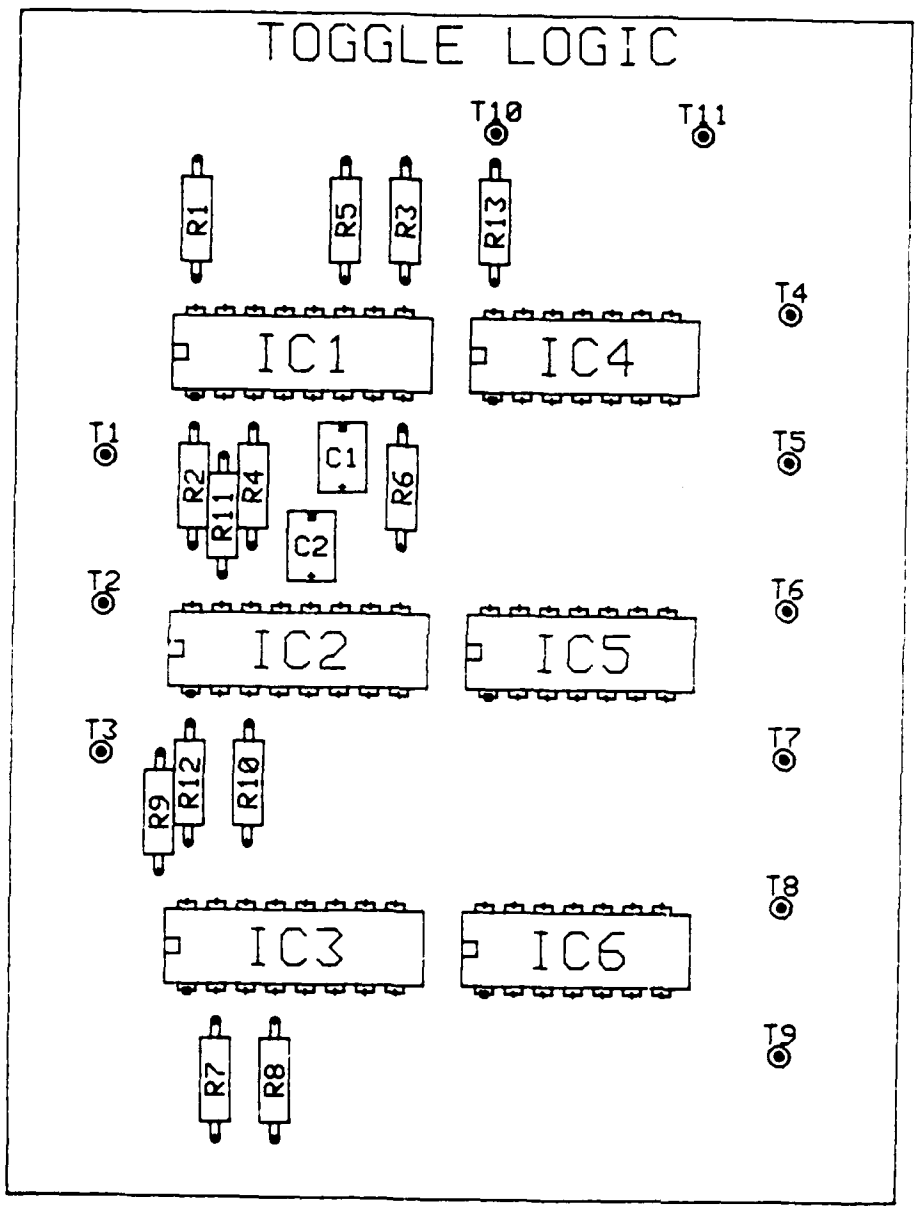


FIG. 2 - A Drawing of Components in Position on a Board.

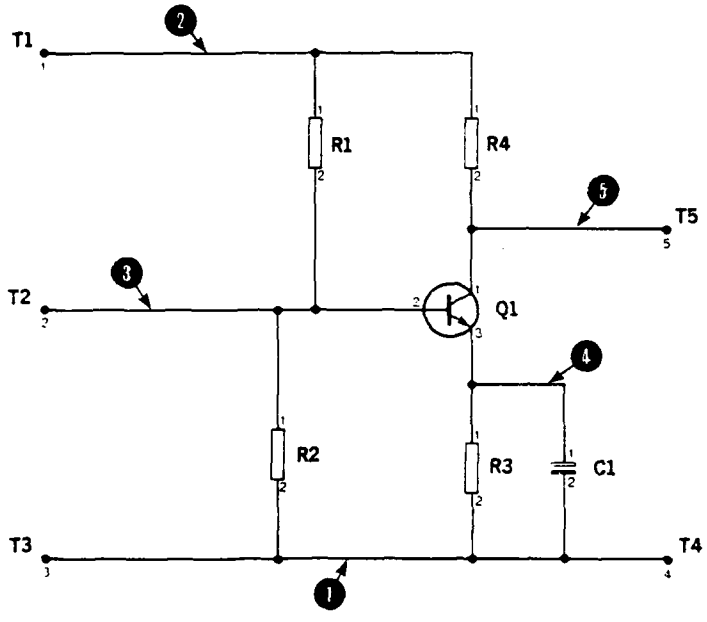
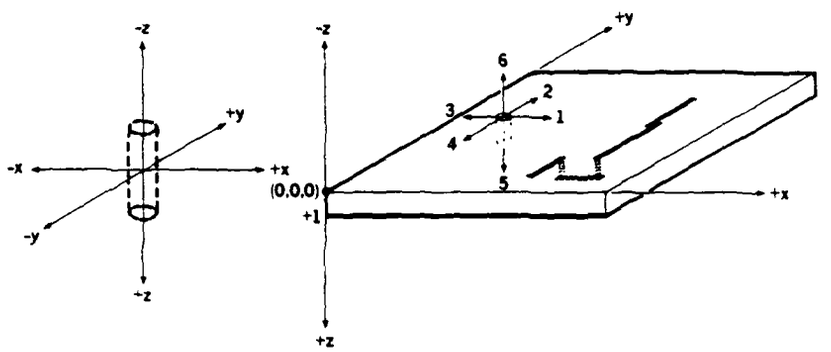


FIG. 3 Assigning Nodes to Encode a Circuit. The node numbers within the circles will be assigned to the terminals of different components.



0 = for Directions other than orthogonal

7 = for Mask Terminus

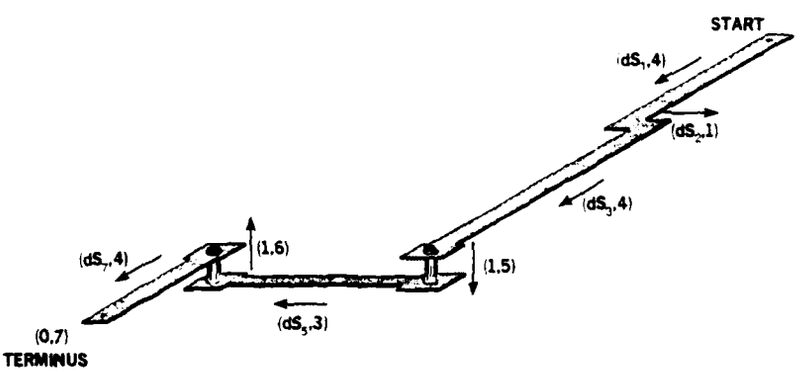


FIG. 4 Illustration of a Typical Mask Construction as Assigned to Link #1 of Node #9 in fig. 5.

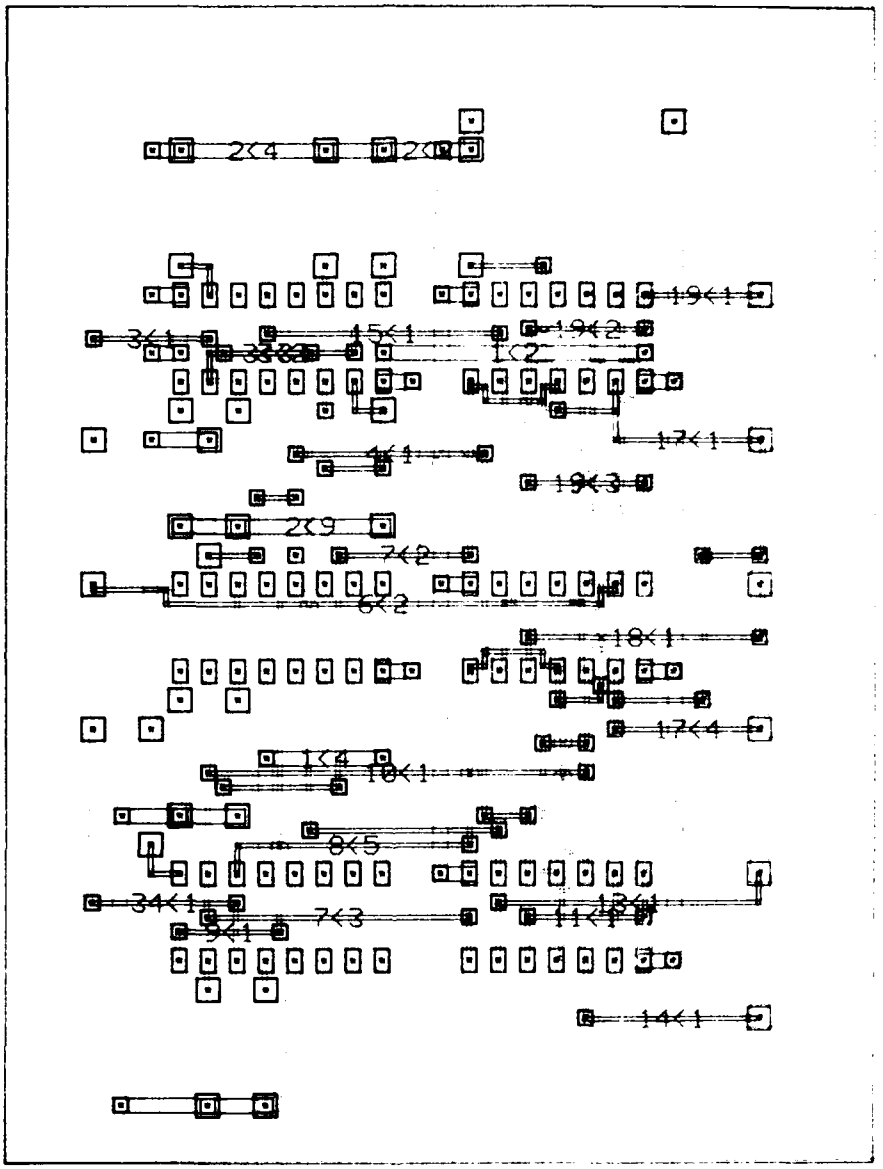


FIG. 5 A Typical Wire-Layout Ready for Inspection after Routing the Board. Links destined for Mask modifications are identified by the labels on them.

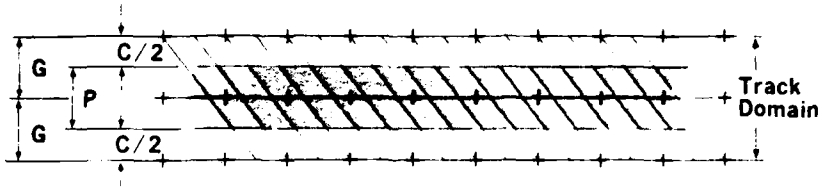


FIG. 6a Representation of a wire-path size of P constructed on a minimum grid increment of size G for clearance C. Actual width of the path of the grid is traced by a 2G x 2G spot, to give a path size P = 2G-C.

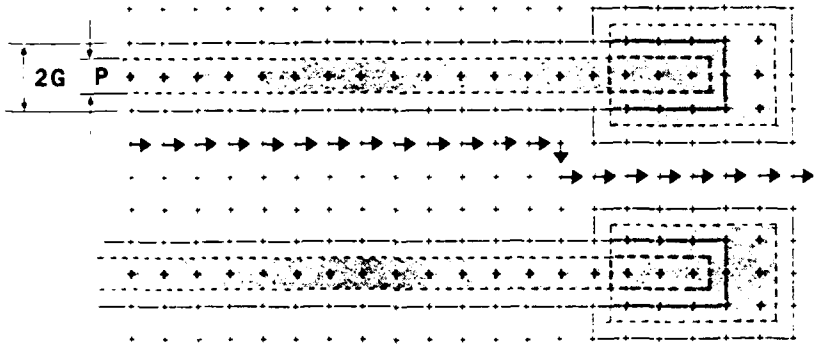


FIG. 6b Twin 2G x 2G Wire-paths terminated in 6G x 4G pads. Arrows show a typical search for a path on the grid using a 2G x 2G spot. Note that the path advances on a single grid-point front even through it is 2 units wide.

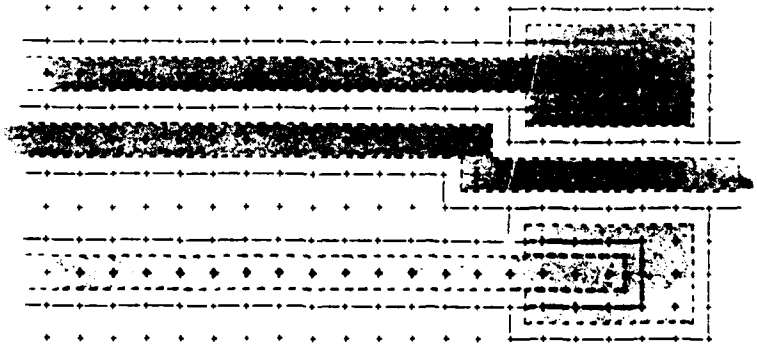


FIG. 6c The search-path of Fig. 6b shown inserted in the grid.

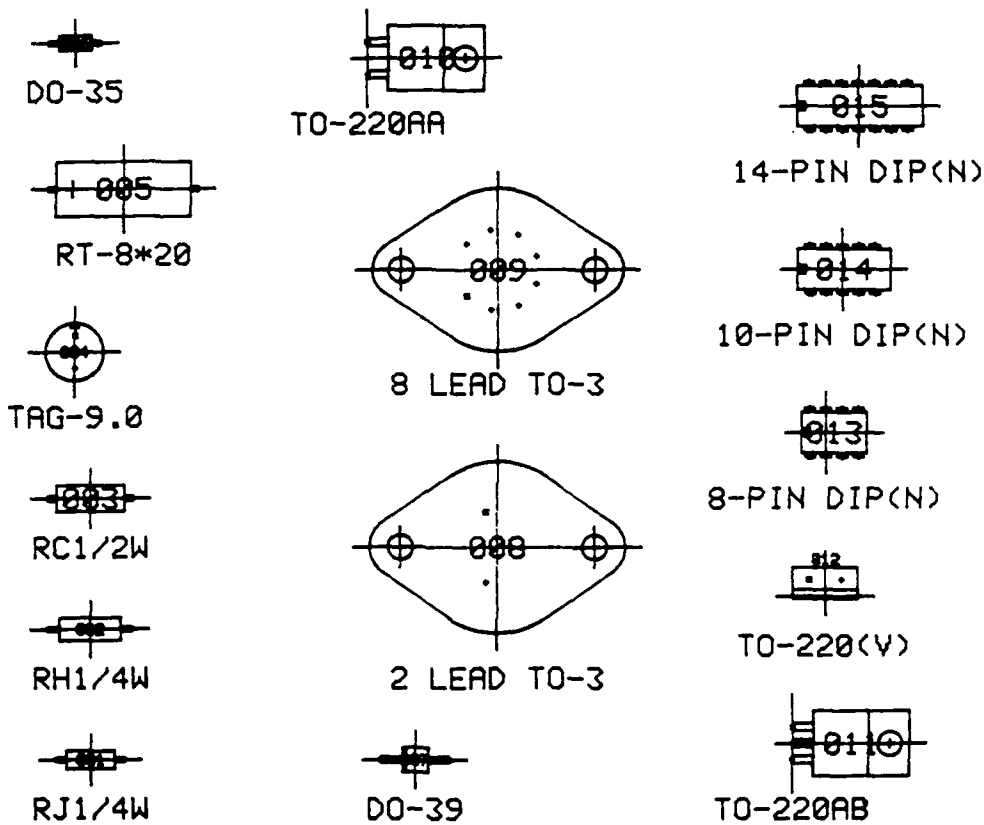


FIG. 7 A small package-outline library constructed as a sub-set of the total library.

	CATALOGUE				PACKAGE		PLACEMENT DATA				
Name	Part Description	G	I	O	Pack name	P	S	F	X	Y	R

Fig. 8a Form for tabulating component declarations prior to keyboard entry.

	PINS (or TERMINALS)																
Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Fig. 8b Form for tabulating (normal) pins prior to keyboard entry.

	DUMMY-PINS 2-LAYER BOARD																	
Name	L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	1																		
	2																		
	1																		
	2																		

Fig. 8c Form for tabulating Dummy-Pins prior to keyboard entry.

END

DATE
FILMED
2-87

DTIC